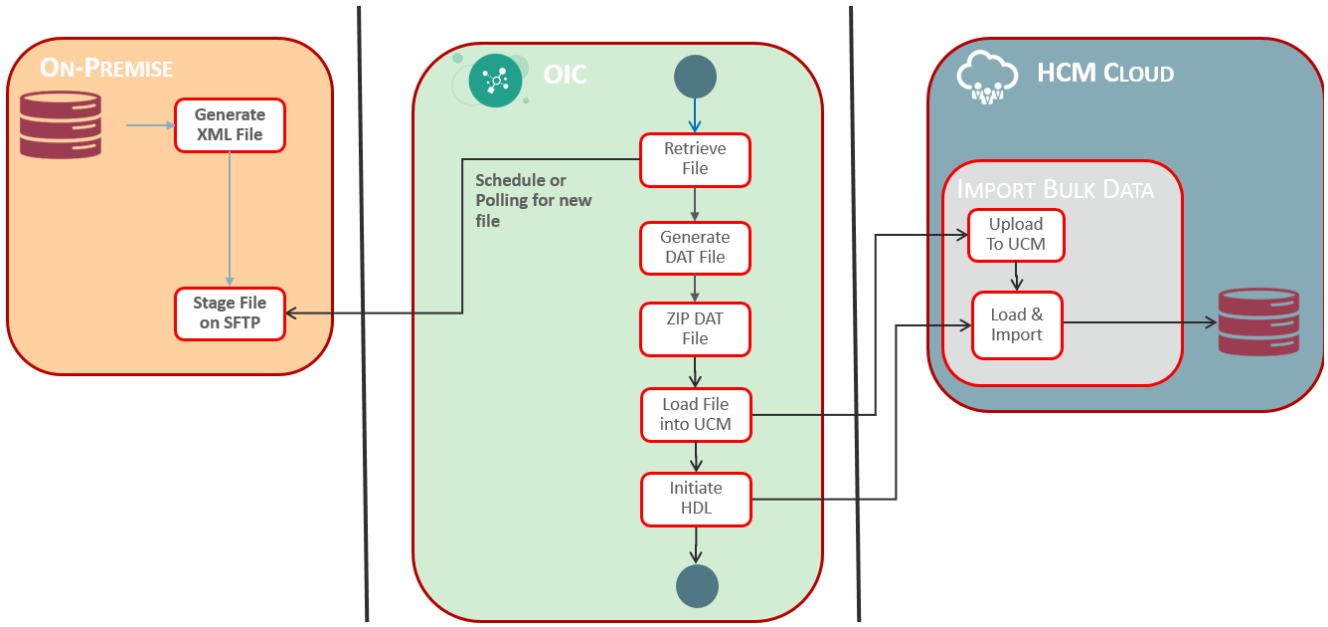
Contents

Executive Summary

Oracle Integration (OIC) provides rich mapping capabilities to transform source data at runtime. Combined with native schema formatting capabilities for output file definition, OIC produces output files in virtually any desired format.

Storyline and Personas

This use case explores the use of OIC to satisfy the needs of reading a file from an external FTP server containing new hires, transforming and formatting the file to generate a valid DAT file. The DAT file is imported into HCM Cloud by the HCM Data Loader to create pending workers. The following diagram illustrates the proposed interaction between the systems involved in this use case.



Objective

This document walks you through some of the mapping steps needed to replicate in your own environment what has been released in the demo title: *Pending Worker Import.*

Constructing the Pending Worker Import

This section works through the steps that are required to build the complete the integration. The integration needs to be imported into your OIC instance prior to the configuration, completion of the integration.

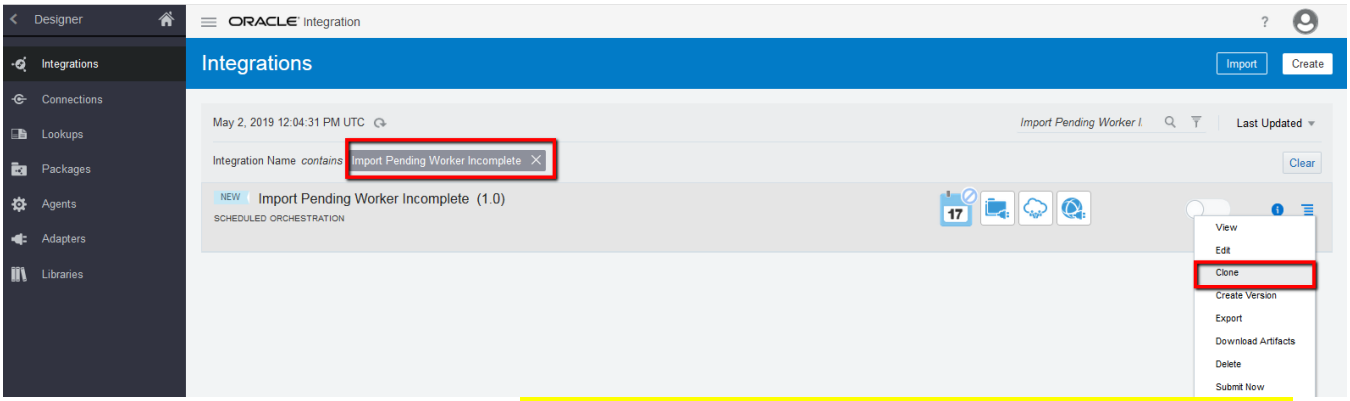
Prerequisites

You will need access to the following applications and artifacts:

- Oracle Integration (OIC)
- HCM Cloud R13+
- FTP Server
- worker-hdl-lab-exercise.nxsd
- persons.xml
- personsSchema.xsd

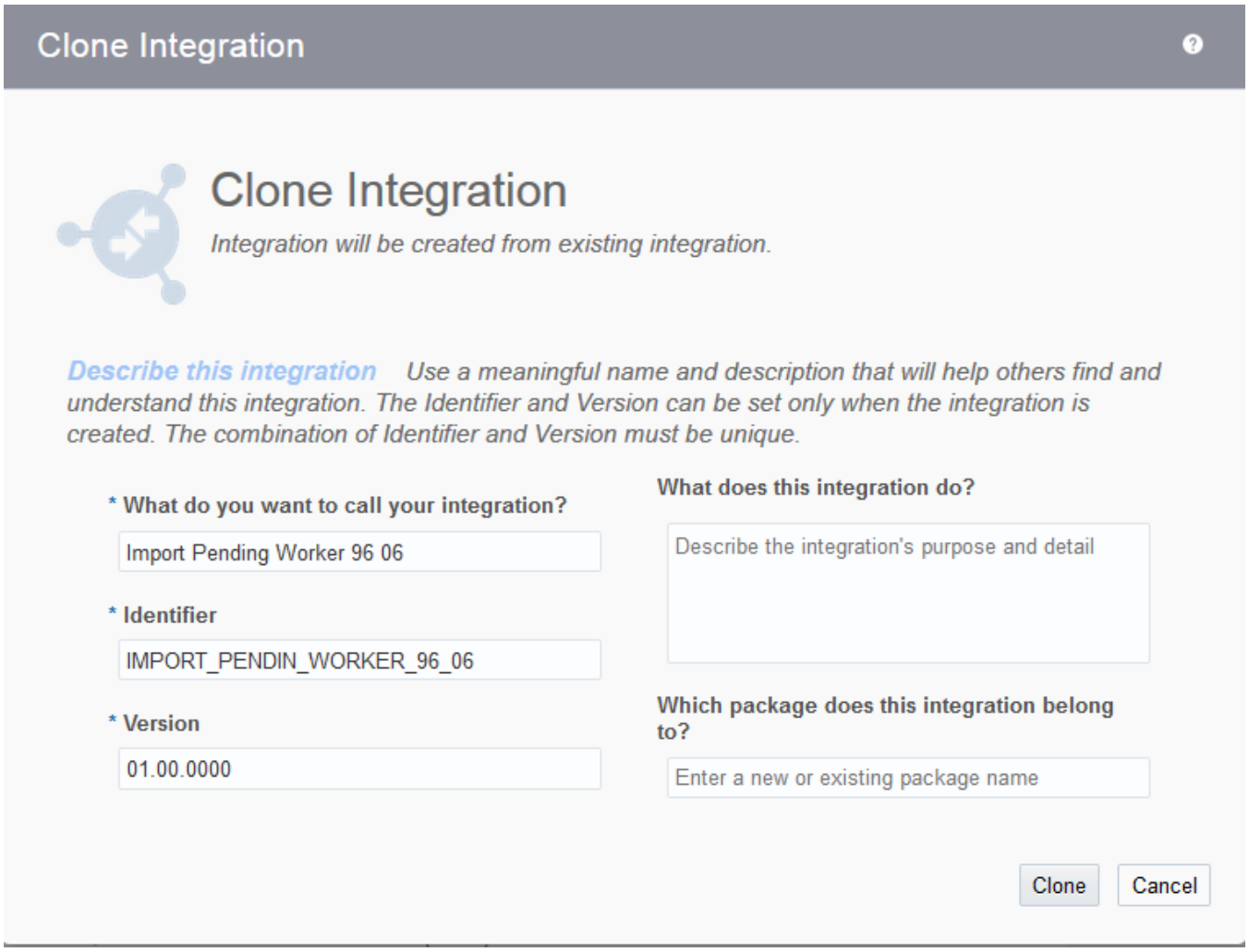
Cloning the Integration

1. Search for the integration “Import Pending Worker Incomplete”. Select “Clone”



Provide a unique name for the Integration flow. Note: Since there are multiple attendees sharing a single environment, please suffix the
<ClassID> <StudentID> to the integration flow

2. E.g. If your <ClassID> is 96 and your <StudentID> is 06, then you will name your Integraiton flow as “Import **Pending Worker 96 06”**



Connections

The following Connections have been created and configured. You will be using these connections for creating Integration flow.

Connection Name	Connection Type
FTP Con 96 06	FTP Adapter
HCM Conn 96 06	HCM Adapter

Background on the Integration (Read Only Section)

HDL DAT File Native Schema

To generating HCM Data Loader compatible DAT files within OIC requires using native NXSD schema. This schema defines the structure of the output file (fields) but also the formatting of data. Using native schema capabilities, you can precisely craft the output format to match the HCM Data Loader needs.

1. Open the accompanying file **worker-hdl-lab-exercise.nxsd** in a text editor.
2. Go to line 3
3. Review the **<schema>** element. It defines the global schema attributes, typically you want to replicate similar <schema> element in every schema you are building for any HDL object only altering the object names.

```
3  <schema xmlns="http://www.w3.org/2001/XMLSchema"
4      xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
5      xmlns:tns="http://supoc.oracle.com/WorkerData"
6      targetNamespace="http://supoc.oracle.com/WorkerData"
7      elementFormDefault="qualified"
8      attributeFormDefault="unqualified"
9
10     nxsd:stream="chars"
11     nxsd:version="NXSD"
12     nxsd:encoding="UTF8"
13     nxsd:encodeLineTerminators="false"
14 >
```

4. Find **<WorkerData>** element. This is the element defines the group of all the objects and child objects that are required by HDL to be loaded for this business object. The example shown here relates to the Worker and several of Worker's child objects.
   Note: two elements are defined for each object:
   - **ObjectLabel** element, used to include the **METADATA** line into the DAT file.
   - **Object** element, used to insert **MERGE** lines into the DAT file.

   The formatting for the lines is also defined, added as METADATA or MERGE lines respectively.

```
16  <element name="WorkerData">
17   <complexType>
18    <choice maxOccurs="unbounded" minOccurs="0">
19     <element ref="tns:WorkerLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|Worker|" />
20     <element ref="tns:Worker" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|Worker|" />
21     <element ref="tns:PersonLegislativeDataLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|PersonLegislativeData|" />
22     <element ref="tns:PersonLegislativeData" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|PersonLegislativeData|" />
23     <element ref="tns:PersonNameLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|PersonName|" />
24     <element ref="tns:PersonName" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|PersonName|" />
25     <element ref="tns:PersonEmailLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|PersonEmail|" />
26     <element ref="tns:PersonEmail" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|PersonEmail|" />
27     <element ref="tns:PersonAddressLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|PersonAddress|" />
28     <element ref="tns:PersonAddress" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|PersonAddress|" />
29     <element ref="tns:PersonPhoneLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|PersonPhone|" />
30     <element ref="tns:PersonPhone" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|PersonPhone|" />
31     <element ref="tns:PersonNationalIdentifierLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|PersonNationalIdentifier|" />
32     <element ref="tns:PersonNationalIdentifier" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|PersonNationalIdentifier|" />
33     <element ref="tns:WorkRelationshipLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|WorkRelationship|" />
34     <element ref="tns:WorkRelationship" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|WorkRelationship|" />
35     <element ref="tns:WorkTermsLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|WorkTerms|" />
36     <element ref="tns:WorkTerms" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|WorkTerms|" />
37     <element ref="tns:AssignmentLabel" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="METADATA|Assignment|" />
38     <element ref="tns:Assignment" minOccurs="0" maxOccurs="unbounded" nxsd:startsWith="MERGE|Assignment|" />
39    </choice>
40   </complexType>
41  </element>
```

5. On inspection of the **WorkerLabel** element. Note this is responsible for adding the proper names of the fields into the **METADATA** line. Each object needs it's corresponding Label object to exist in the schema.
   Worker has **WorkerLabel**, **Assignment** has **AssignmentLabel** etc.

```
<element name="WorkerLabel">
  <complexType>
    <sequence>
      <element name="EffectiveStartDateLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="EffectiveEndDateLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="StartDateLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="CategoryCodeLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="ActionCodeLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="DateOfBirthLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="SourceSystemOwnerLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="SourceSystemIdLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="ApplicantNumberLabel" type="string" nxsd:style="terminated" nxsd:terminatedBy="${eol}" />
    </sequence>
  </complexType>
</element>
```

6. Review the **<name=Worker>** element. This section defines all the fields required by HDL's Worker object. Formatting definition are defined with separators and ends of the lines.

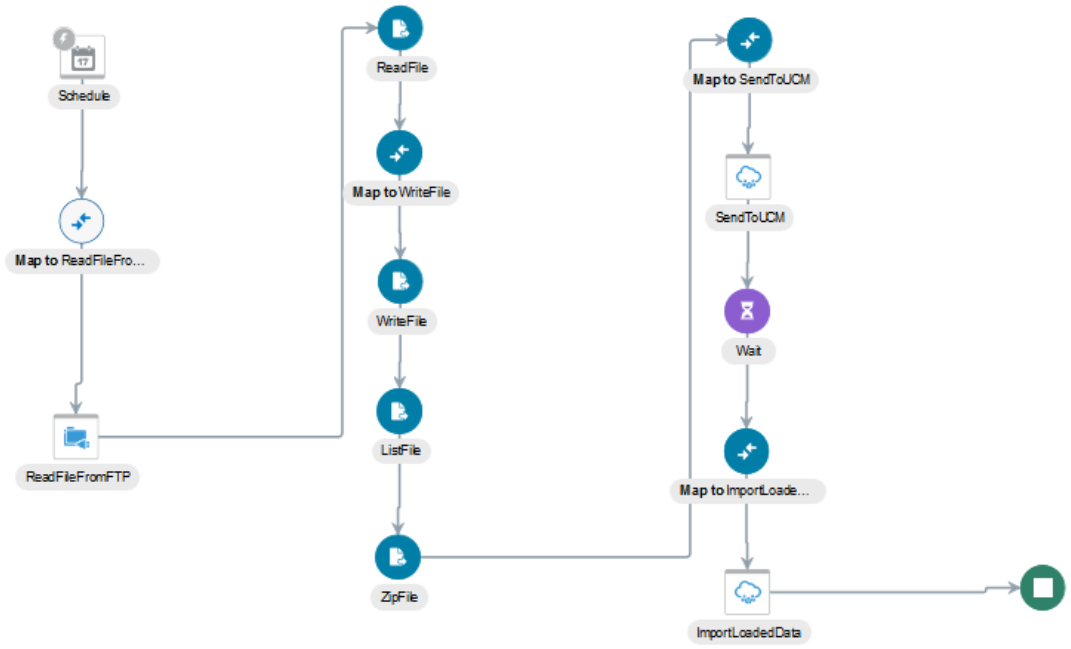
```
<element name="Worker">
  <complexType>
    <sequence>
      <element name="EffectiveStartDate" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="EffectiveEndDate" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="StartDate" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="CategoryCode" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="ActionCode" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="DateOfBirth" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="SourceSystemOwner" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="SourceSystemId" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="ApplicantNumber" type="string" nxsd:style="terminated" nxsd:terminatedBy="${eol}" />
    </sequence>
  </complexType>
</element>
```

7. Review the **<element name="PersonName">** element. This section defines all the fields required by HDL's Person object. Formatting definition are defined with separators and ends of the lines.

```
<element name="PersonName">
  <complexType>
    <sequence>
      <element name="PersonIdSourceSystemId" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="EffectiveStartDate" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="EffectiveEndDate" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="LegislationCode" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="NameType" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="FirstName" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="MiddleNames" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="LastName" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="SourceSystemOwner" type="string" nxsd:style="terminated" nxsd:terminatedBy="|" />
      <element name="SourceSystemId" type="string" nxsd:style="terminated" nxsd:terminatedBy="${eol}" />
    </sequence>
  </complexType>
</element>
```

Integration Overview

On the Oracle Integration Cloud Service home page, click the **Import Pending Worker Import XX XX** integration name. This will open the integration displaying the various steps to fulfil the integration.



The follow table describes the integration at a high level. For more detail on each step, please open and review from within the integration.

Step	Type	Description
Schedule	Schedule	Beginning of the integration.
Map to ReadFileFromFTP	Map To	
ReadFileFromFTP	FTP Adapter: Download File	
ReadFile	Stage File: Read	Reads the results of the file retrieved from the FTP server into a defined schema
Map to WriteFile	Map To	Performs the data mapping and transformation generating the HDL DAT file
WriteFile	Stage File: Write Files	Write the results of the data mapping and transformation into a virtual directory within OIC
ListFile	Stage File: List Files	Returns a list of files matching the a file name pattern from the virtual directory within OIC
ZipFile	Stage File: Zip Files	Zips the worker.dat file which is written as part of the “Write File” operation
Map to SendToUCM		Maps the required UCM properties to Check in the file into UCM
SendToUCM		“CheckIn” the file into UCM using HCM adapter
Wait		2 Minutes Wait Time
Map to ImportLoadData		Map the ContentId uploaded to UCM successfully
ImportLoadData		Uploads the file into HCM cloud using the UCM ContentId
Stop	Stop	Denotes the end of the integration

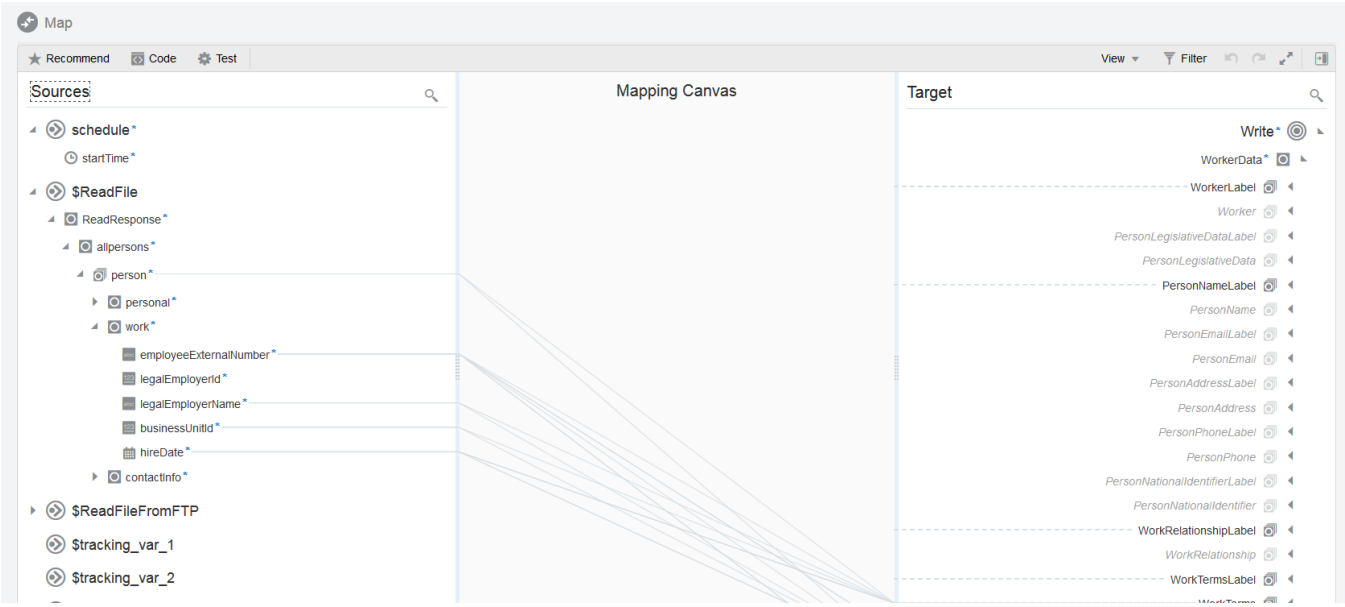
Edit ReadFileFromFTP (FTP Adapter: Read a File)

This task defines the location and name of the file to be picked up. This needs to be edited to reflect your FTP server instance.

1. Click the FTP adapter icon (**ReadFileFromFTP)**.
2. Click **Edit** to open the FTP Adapter configuration Wizard page.
3. Select the **Operations** tab
4. Update the FTP directory value for the **Input Directory**. This should reflect the location of where you have uploaded your copy of the persons.xml file. (Note: Configure a dedicated ftp folder ex: /upload/public_ftp/XX where "XX" is the name of your folder)
5. Select **Next**. This will show the Schema tab.
6. Select **Next.** This will show the **Summary tab.**
7. Select **Done.** This will close the **FTP Adapter** configuration wizard and open the **Update Configuration?** dialogue window.
8. Select **Update**. This will close the **Update Configuration?** dialogue window, returning you to the integration canvas.

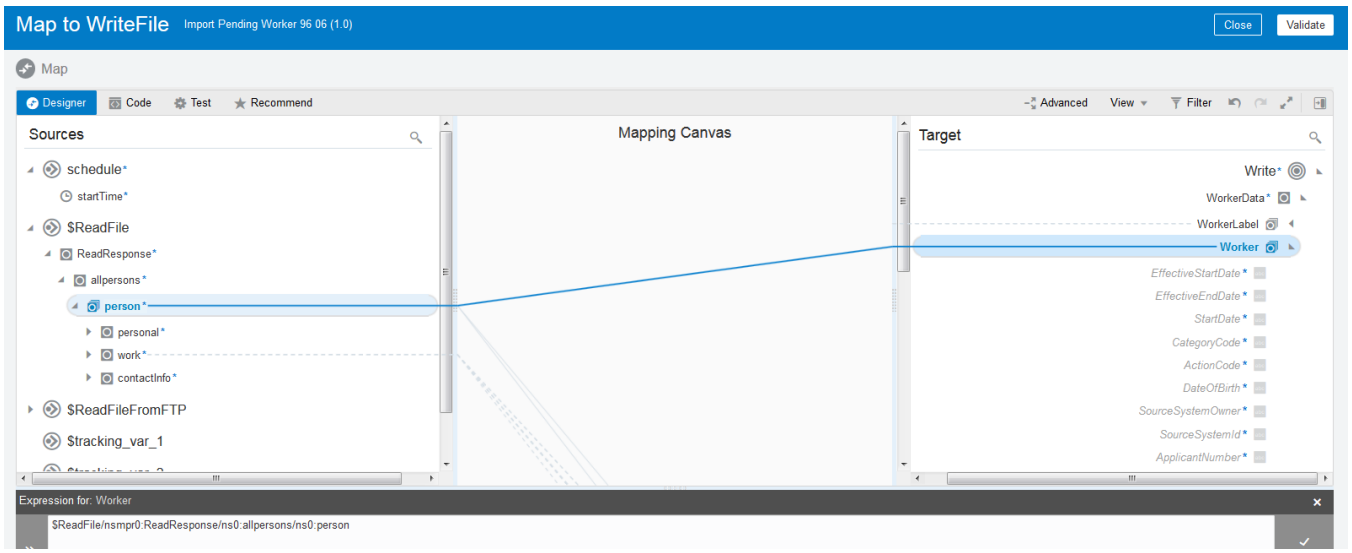
Edit Map To WriteFile

1. Click the Mapper icon (**Map To WriteFile)**.
2. Click **Edit** to invoke the mapper.
   The left hand panel **Sources,** shows all of the available values and fields that can used in this mapping. The **Target** panel shown on the right illustrates the **Write** hierarchy. This is a representation of the basic structure for a **Worker.dat** file.

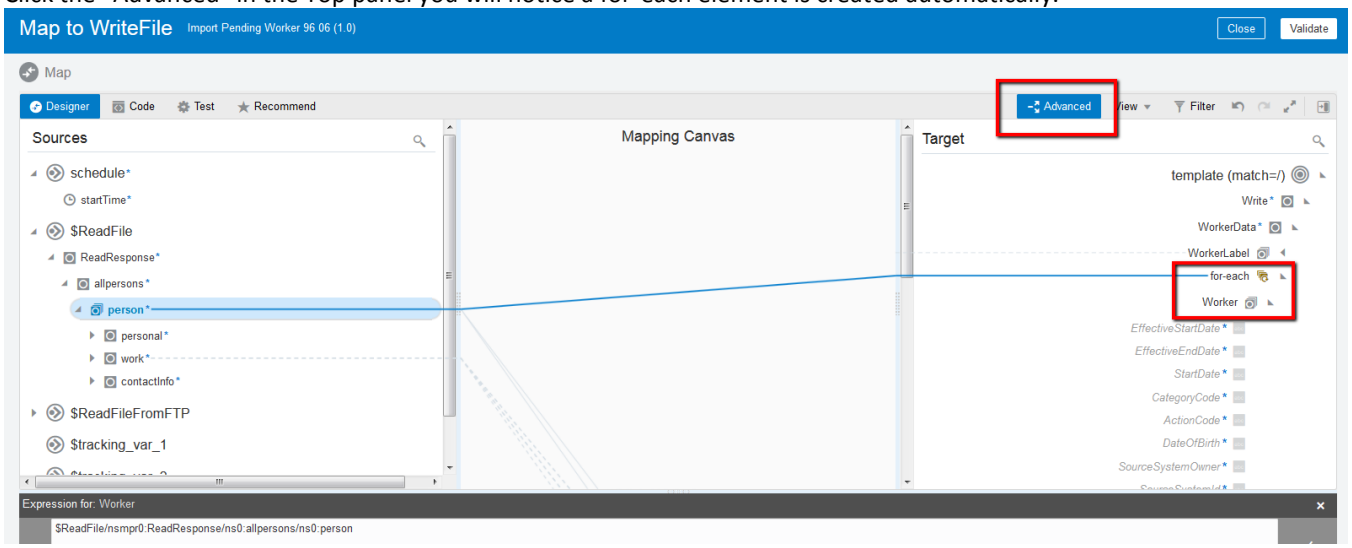


Several mapping elements have been completed for you, however it is necessary to define several remaining values.

3. From the **Sources** panel, select the repeating element **Person** field and connect to the repeating element **Worker** on the Target panel.



Click the “Advanced" in the Top panel you will notice a for-each element is created automatically.



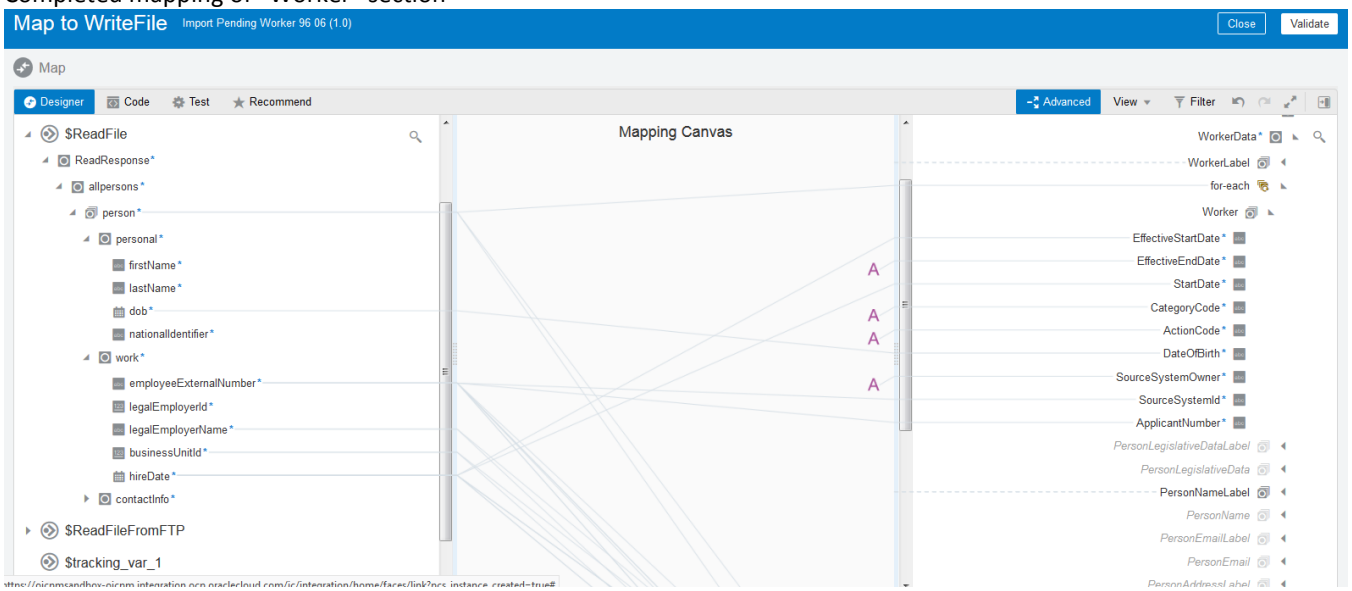
Worker hierarchy is now linked with **Person** hierarchy; and **Worker** is a child of for-each loop. This means: for each **Person** record in source, a target **Worker** record will be created.

Worker

The following table is an overview of the mapping for the **Worker** section. Perform the mapping activity as per the source and target element specified in the following table.

Source	action	Target
ReadFile/ReadResponse/allpersons/person/work/hireDate	Map field to	Worker/EffectiveStartDate
'4712/12/31'	Assign value	Worker/EffectiveEndDate
ReadFile/ReadResponse/allpersons/person/work/hireDate	Map field to	Worker/StartDate
'M'	Assign value	Worker/CategoryCode
'ADD_PEN_WKR'	Assign value	Worker/ActionCode
ReadFile/ReadResponse/allpersons/person/personal/dob	Map field to	Worker/DateOfBirth
'HRC_SQLLOADER'	Assign value	Worker/SourceSystemOwner
ReadFile/ReadResponse/allpersons/person/work/ employeeExternalNumber	Map field to	Worker/SourceSystemId
ReadFile/ReadResponse/allpersons/person/work/ employeeExternalNumber	Map field to	Worker/ApplicantNumber

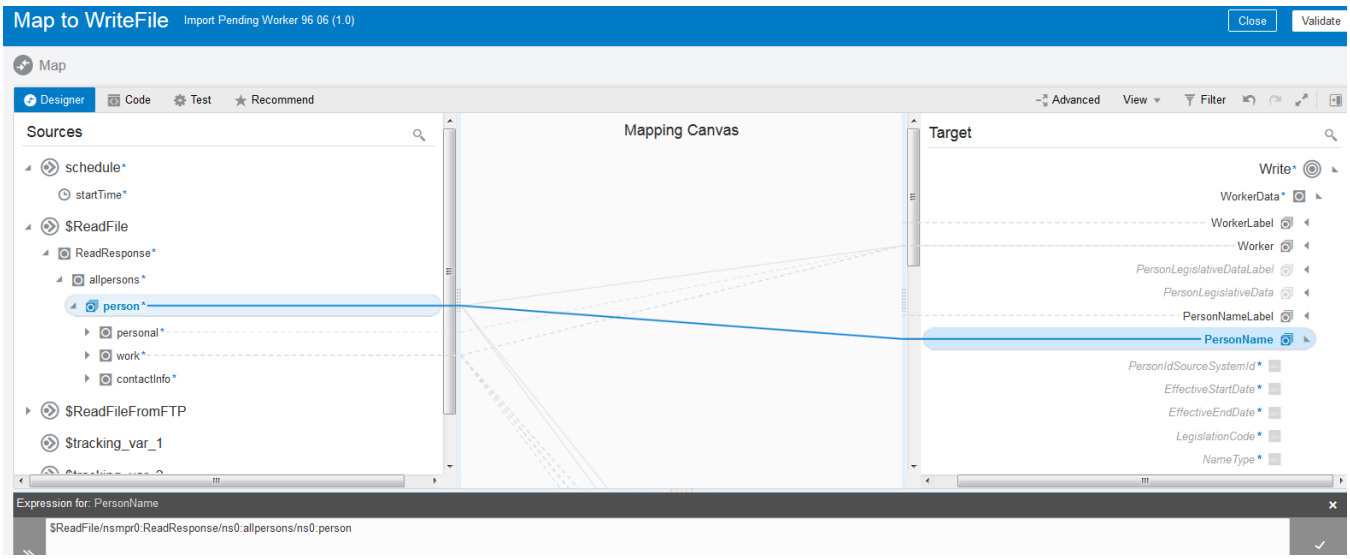
Completed mapping of “Worker” section



Save your mapping.

PersonName

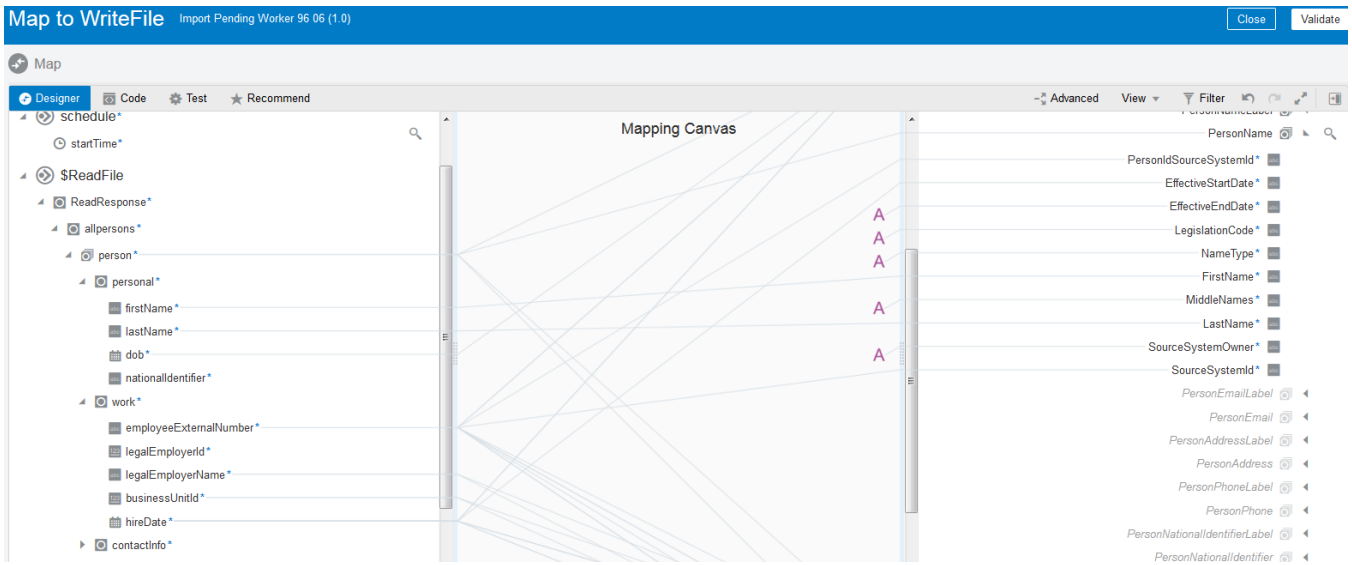
4. From the **Sources** panel, select the repeating element **Person** field and connect to the repeating element **PersonName** on the Target panel.



Person hierarchy is now linked with **PersonName** hierarchy; and **PersonName** is a child of a for-each loop. This means: for each **Person** record in source, a target **PersonName** record will be created.

The following table is an overview of the mapping for the **PersonName** section. Perform the mapping activity as indicated in the table below.

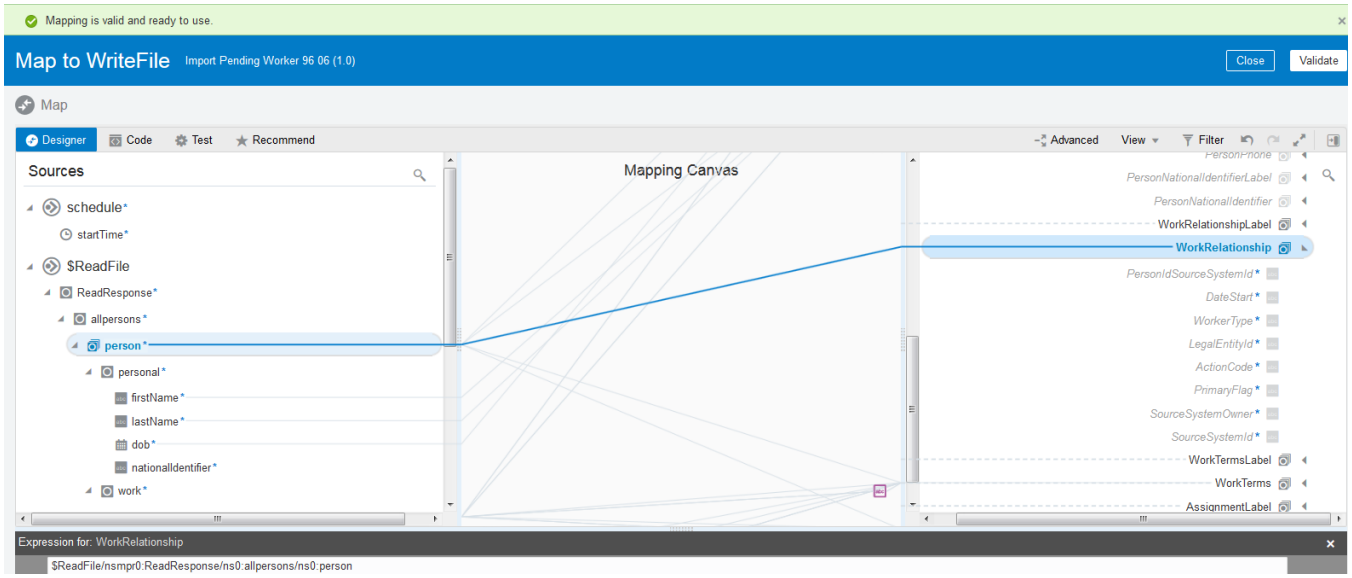
Source	**action**	**Target**
ReadFile/ReadResponse/allpersons/person/work/ employeeExternalNumber	Map the field	PersonName/personIDsourceSystemID
ReadFile/ReadResponse/allpersons/person/work/ hireDate	Map the field	PersonName/EffectiveStartDate
'4712/12/31'	Assign value	PersonName/EffectiveEndDate
'US'	Assign value	PersonName/LegislationCode
'GLOBAL'	Assign value	PersonName/NameType
ReadFile/ReadResponse/allpersons/person/personal/ firstName	Map the field	PersonName/FirstName
''	Assign value	PersonName/MiddleNames
ReadFile/ReadResponse/allpersons/person/personal lastName	Map the field	PersonName/LastName
'HRC_SQLLOADER'	Assign value	PersonName/SourceSystemOwner
ReadFile/ReadResponse/allpersons/person/work/ employeeExternalNumber	Map the field	PersonName/SourceSystemId



5. Save your mapping.

WorkRelationship

6. From the **Sources** panel, select the repeating element **Person** field and connect to the repeating element **WorkRelationship** on the Target panel.

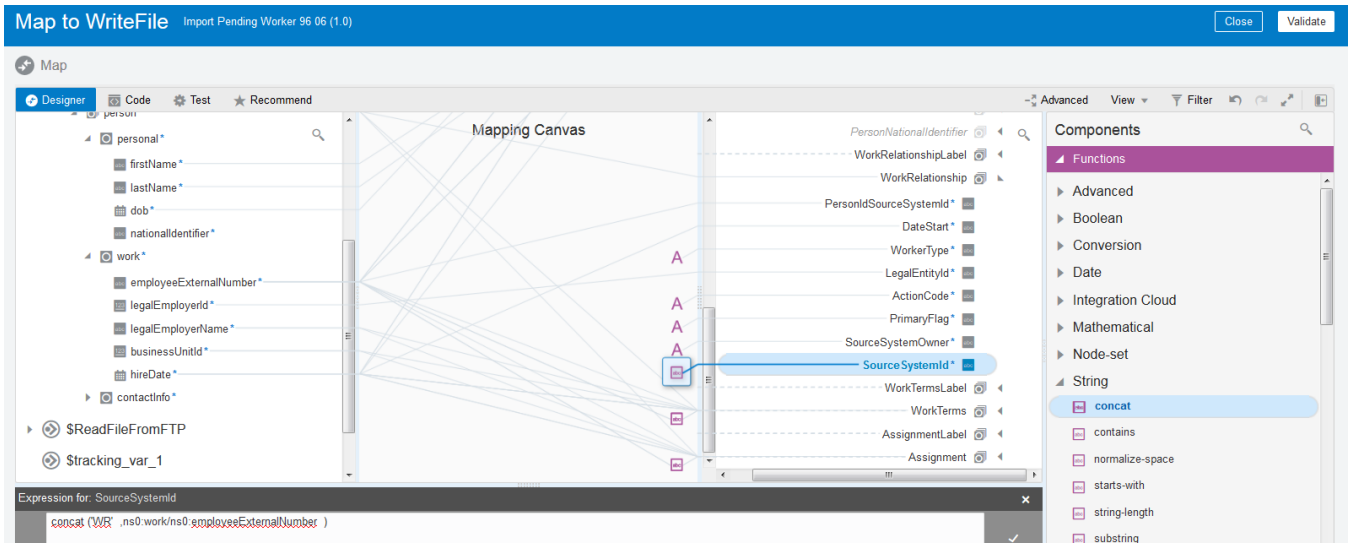


Person hierarchy is now linked with **WorkRelationship** hierarchy; and **WorkRelationship** is a child of a for-each loop. This means: for each **Person** record in source, a target **WorkRelationship** record will be created.

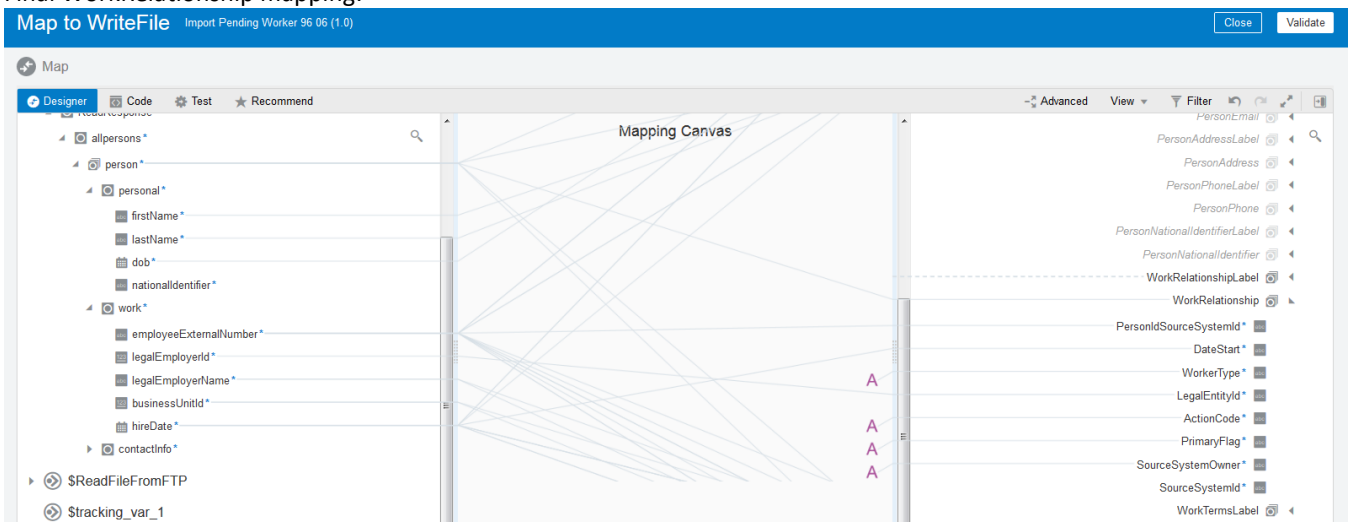
The following table is an overview of the mapping for the **PersonName** section. Perform the mapping activity as indicated in the table below.

Source	action	Target
work/employeeExternalNumber	Map the field	WorkRelationship /personIDsourceSystemID
work/hireDate	Map the field	WorkRelationship /DateStart
'P'	Assign value	WorkRelationship /WorkerType

work/legalEmployerID	Map the field	WorkRelationship /LegalEntityID
'ADD_PEN_WKR'	Assign value	WorkRelationship /ActionCode
'Y'	Assign value	WorkRelationship /PrimaryFlag
'HRC_SQLLOADER'	Assign value	WorkRelationship /SourceSystemOwner
'WR' + work/employeeExternalNumber	Use concatenate function	WorkRelationship /SourceSystemID



Final WorkRelationship mapping:



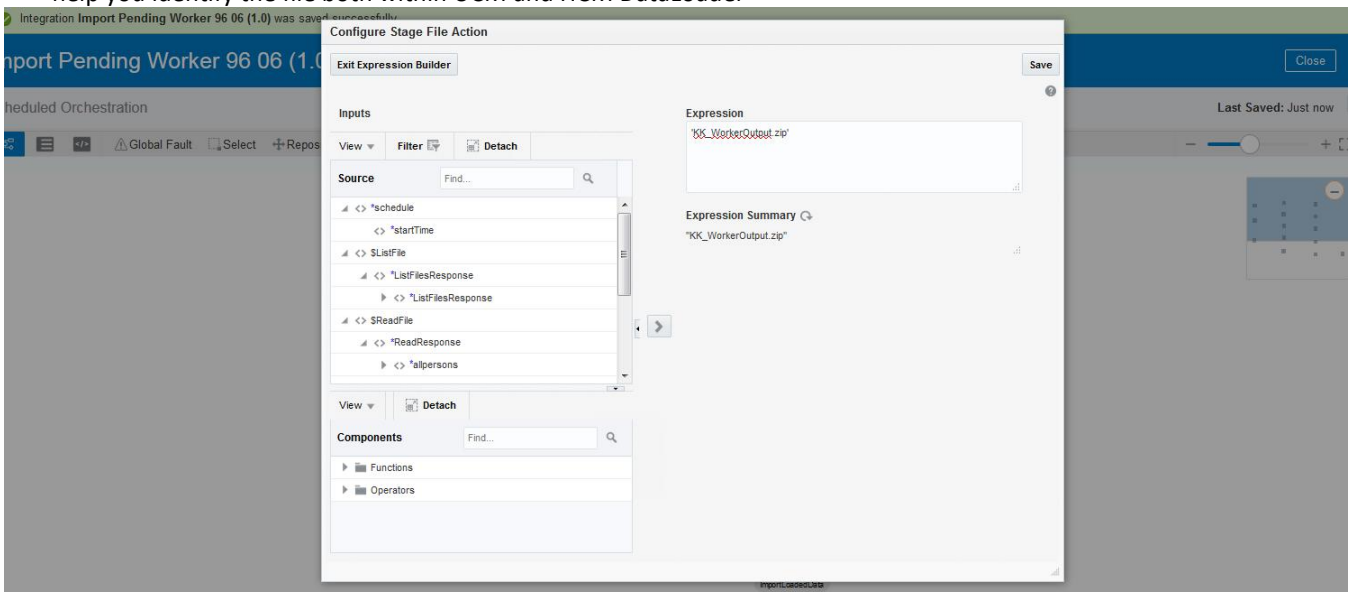
Save your mapping.
The SourceSystem ID must be specially designed for WorkTerms to distinguish entries for WorkTerms, WorkRelationship and Assignment. By adding WR you are able to make this distinction.

The remainder of the mapping has been completed for you. Please review and understand the mapping values that have been used.

Edit ZipFile (Stage File Action)

This task zips and defines a name for the data file being loaded into UCM.

1. Click the Stage file Action "**ZipFile"**
2. Click **Edit** to open the **Stage File Action** configuration Wizard page.
3. Select the **Configure Operation** tab
4. Select the pencil icon for the **Specify the File name**
5. Edit the **Expression** value to include your initials at the beginning of the file e.g. `KK_WorkerOutput.zip`. This will help you identify the file both within UCM and HCM DataLoader

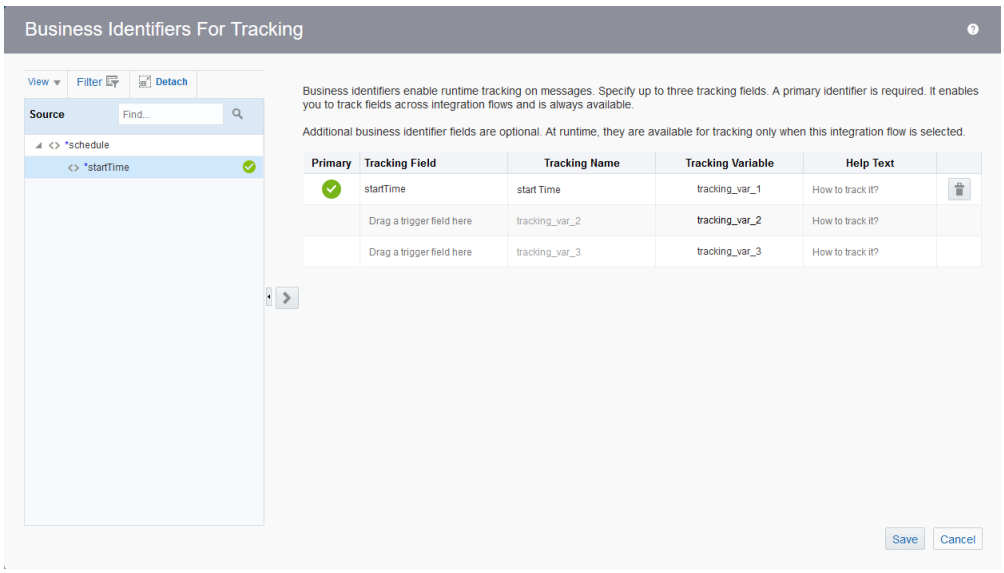


6. Select **Next.** This will show the **Summary tab.**
7. Select **Done.** This will close the **Stage File Action** configuration wizard and open the **Update Configuration?** dialogue window.
8. Select **Update**. This will close the **Update Configuration?** dialogue window, returning you to the integration canvas.
9. Select Save

Set the Business Identifier For Tracking.

The final step in configuring the integration is the setting of the tracking identifier. This is used to provide a unique value for monitoring and reporting.

1. From the action menu on the integration canvas, select the **Tracking** option. This will open the **Business Identifiers for Tracking** dialogue window
2. Drag the **startTime** value from the Source panel on to the first **Tracking** field.
3. Select **Save**. This is will close the **Business Identifiers for Tracking dialogue** window and return you to the integration canvas.

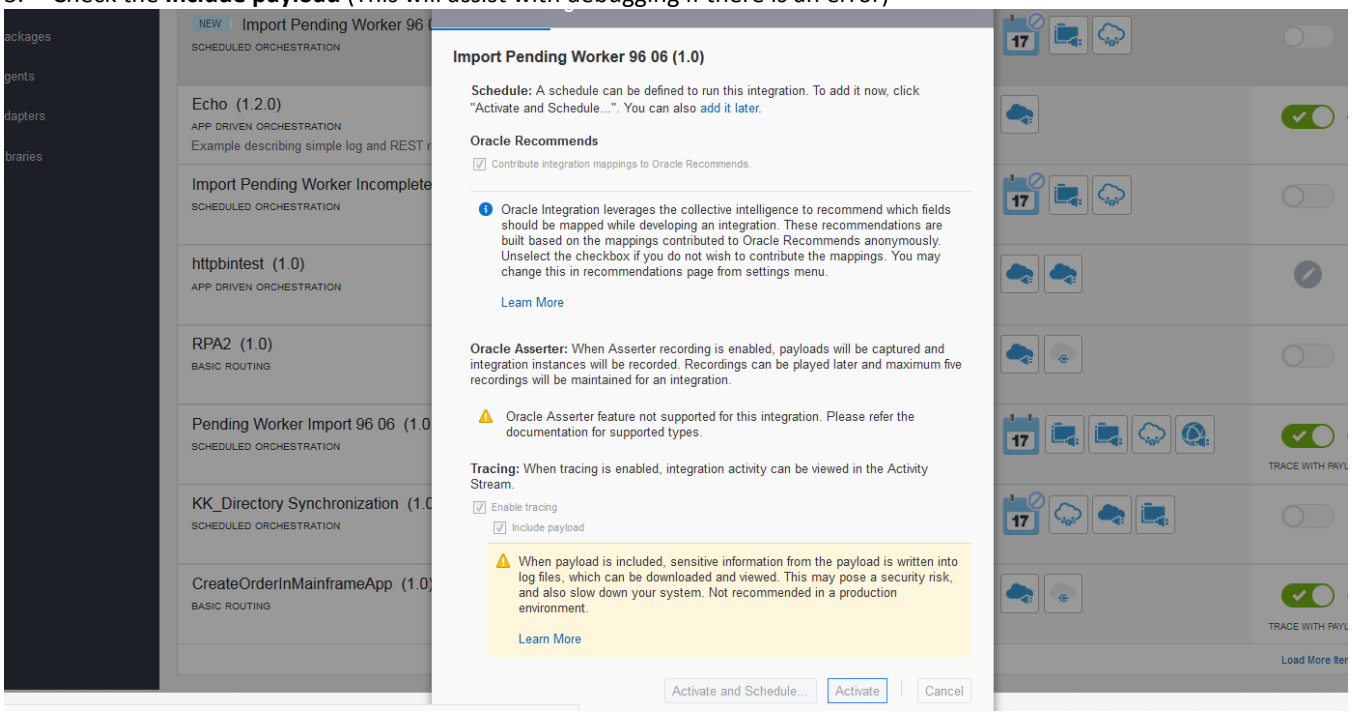


4. Select **Save**.

   You are now ready to activate and execute the integration.

Activate and Run the Integration.

1. Find your integration flow on the list and click the **Activation switch**. This will open the **Activate Integration** dialogue window.
2. Check the **Enable tracing** (This will assist with debugging if there is an error)
3. Check the **Include payload** (This will assist with debugging if there is an error)



4. Select the **Activate** button
5. Once the **Activation** is complete, select from the action menu the **Submit Now** action.
6. This will execute the integration. A **request id** value will be displayed.
7. Click the **request id** value. This will open the **Monitor Runs** page. Here you can view the execution of the integration.

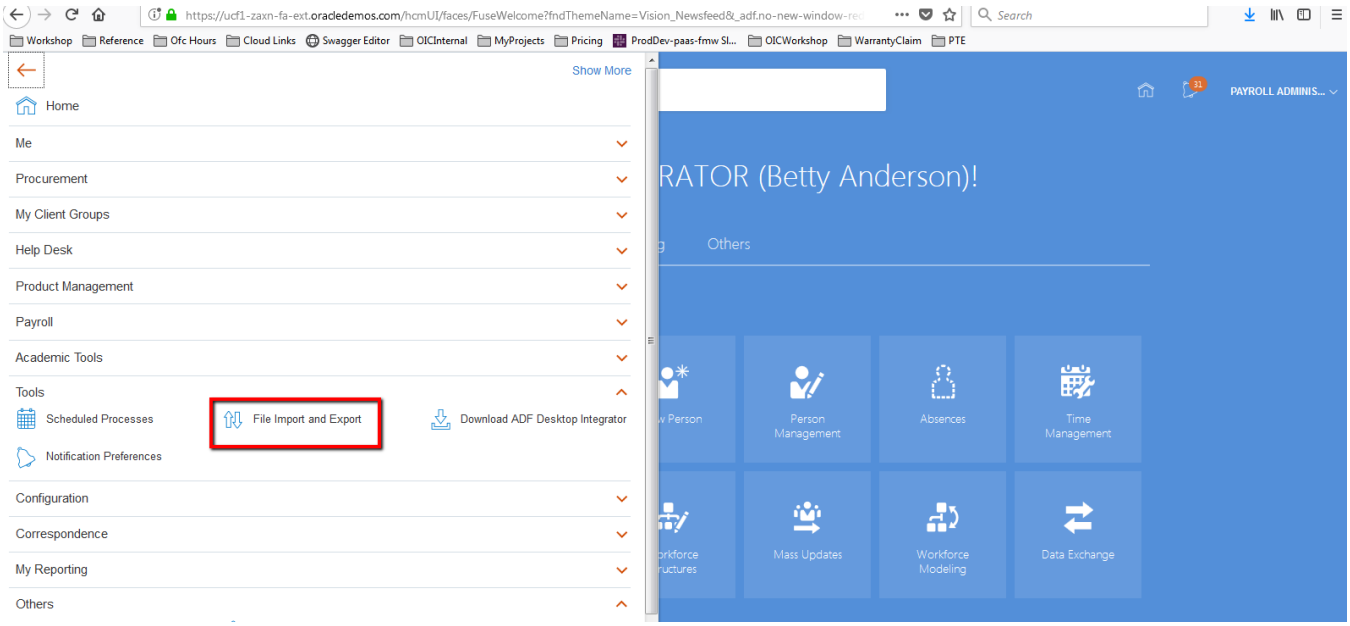
Remember your integration will take 2+ minutes to execute as we have configured “Wait” activity for 2 minutes ☺
Once the integration has completed a Successful notification will be displayed.

8. Select the **RUN ID** value. This will open the Tracking summary page for this integration.
9. Select the **start Time:** value (which is the **Business Identifier** set on the integration earlier). This will open the integration instance screen, displaying all of the steps within the integration. They will be green, indicating that they were successful.

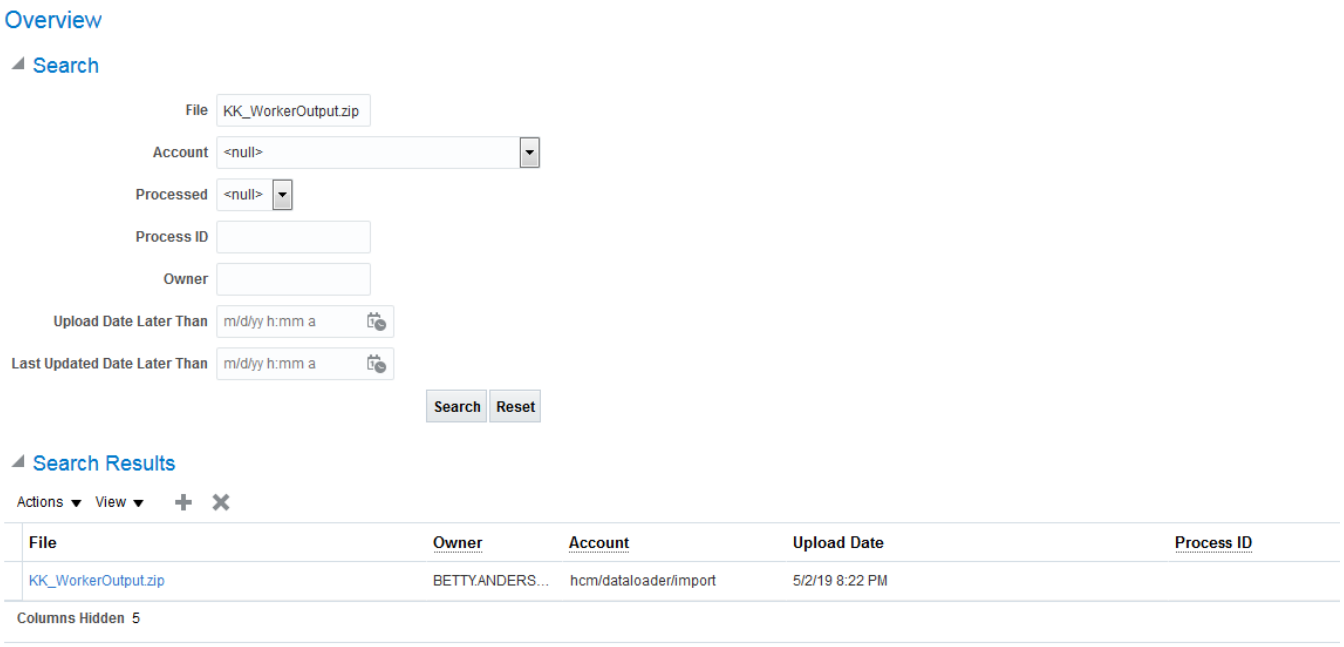
Validating File presence in UCM

During the execution of the integration it is possible to validate the initial transfer of the file to **UCM** prior to the import and loading by **HCM Data Loader**. This is achieved in the following way.

1. Login to your assigned HCM Cloud instance as “betty.anderson”
2. Open the **Navigator**
3. Select the **File Import and Export** link. This will open the **File Import and Export** window.



4. Enter the file name (XX_Worker.dat) (Replace "XX" with your initials provided in the earlier steps) as search criteria.
5. Select **Search**. The **Search Results** panel should display the file loaded into **UCM**.

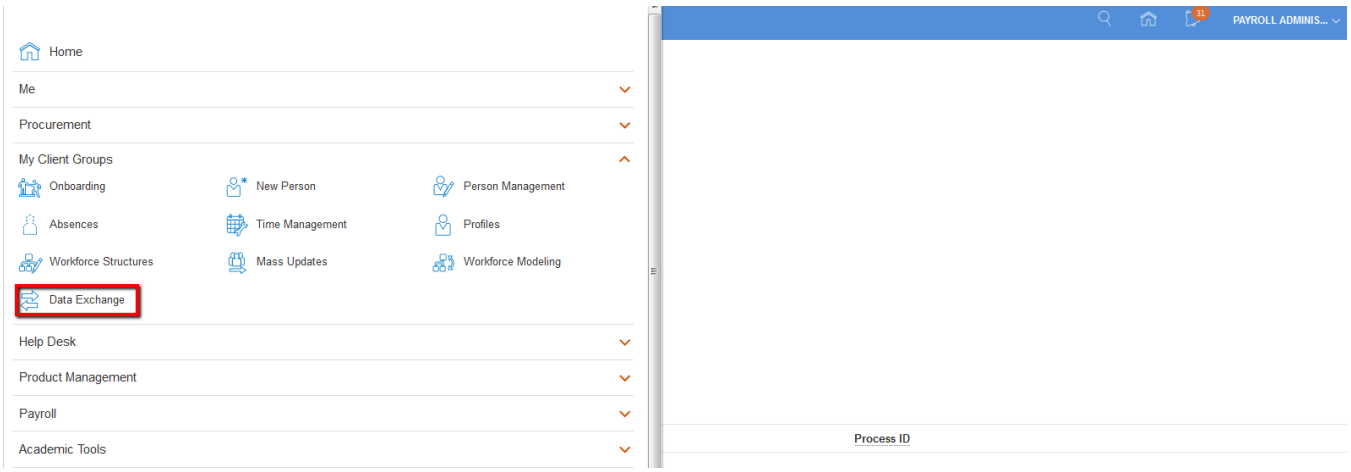


The file will only be displayed until the Import and Load action has completed. Once the **HCM Data Loader** has completed, the file will be automatically deleted from **UCM** and no longer appear when searching via the **File Import and Export** page.

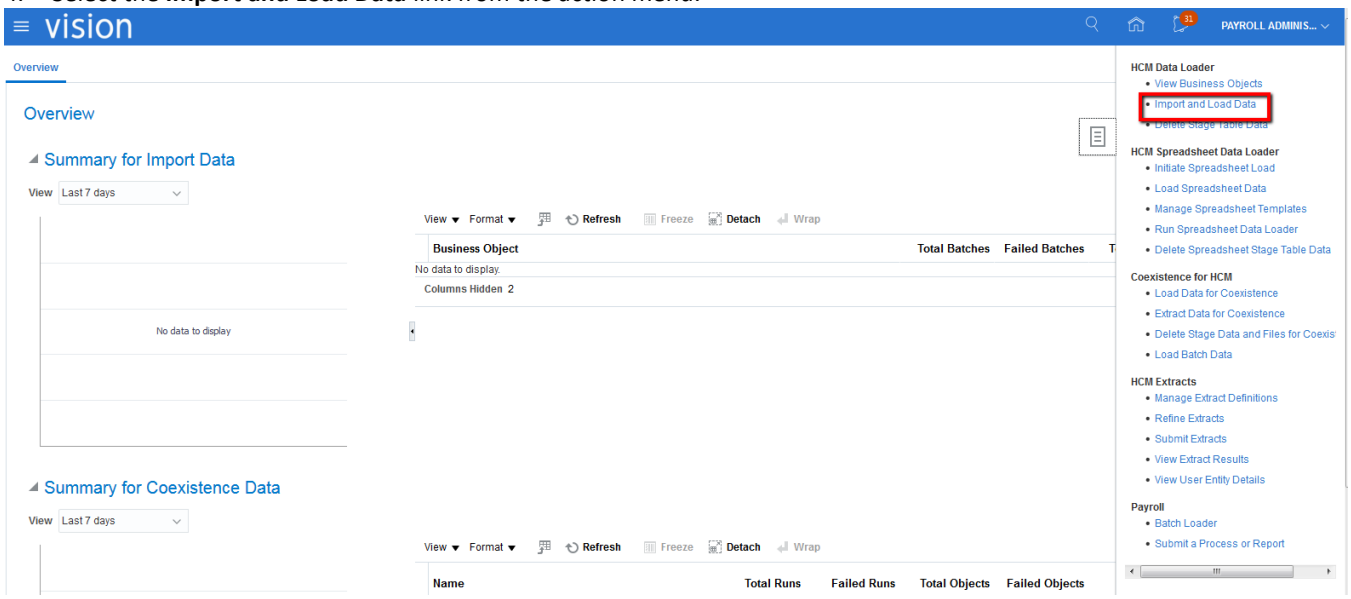
Validating Data Import and Load in HCM Data Loader

Next you can validate the file has been retrieved and the data loaded into HCM Cloud via the HCM Data Loader. This can be achieved in the following way.

1. Open the **Navigator**
2. Select the **Data Exchange** link. This will open the **Data Exchange** window.



3. On the **Data Exchange** window, select the action menu on the right side.
4. Select the **Import and Load Data** link from the action menu.

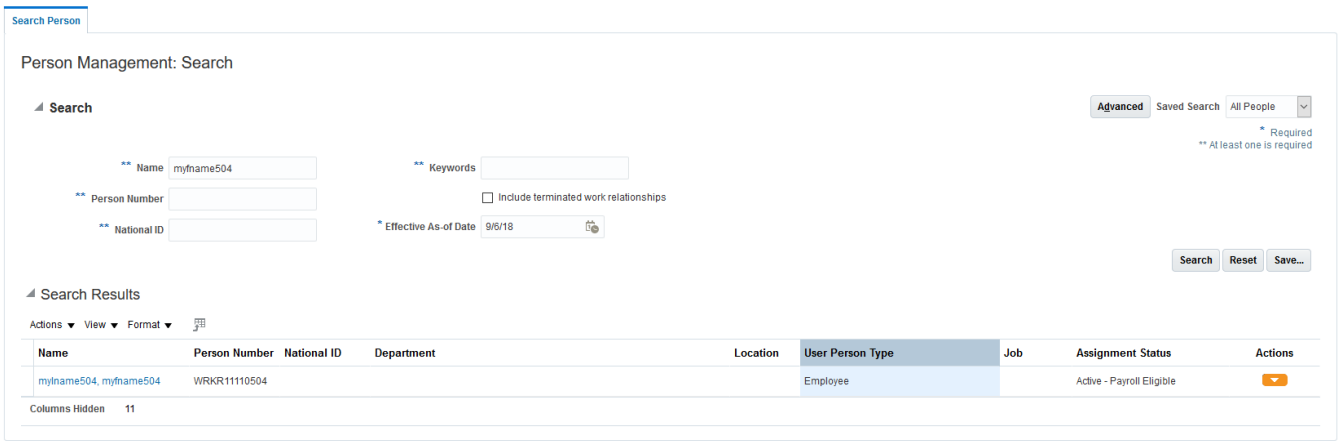


This will open **the Import and Load Data** window.

The lower section of the **Import and Load Data** window shows the current data sets being processed. The **Data Set** column will display the name of the file retrieved from the FTP Server.

5. On the **Person Management**: Search page, enter in the Name field the value of the an employee from the original persons.xml file loaded on the your FTP server
6. Select Search.

The Search Results will display the full name of the worker and additional information. This confirms that the file has successfully imported into your HCM Cloud instance.



Congratulations☺, you have now completed an end-to-end Flow to fetch, transform and automate HCM Cloud HDL import of Pending Workers.